REMARKS

Reconsideration of the above-identified patent application in view of the amendments above and the remarks following is respectfully requested.

Claims 1-23 are in this case. Claims 1, 2, 10, 12, 13, 16, 18, 20 and 21 have been rejected under § 102(e). Claims 3-9, 11, 14, 15, 17, 19, 22 and 23 have been rejected under § 103(a). Independent claims 1, 12 and 18-22 and dependent claim 23 have been amended. Dependent claim 2 has been canceled. New independent claims 24-28 have been added.

The claims before the Examiner are directed toward a flash memory device for storing code to be executed by a processor. The device includes a flash memory, for storing the code, such that the code cannot be executed in place from the flash memory. The device also includes a volatile memory to which the code is copied for execution, and a logic, separate from the processor, for doing the copying. If the code is boot code, the device can be used to boot a system that includes the processor.

Please note that the numbering of the claims herein is in conformity with the numbering in the patent application as published (publication no. 2003/0028708). (In the patent application as filed, claims 19-23 were inadvertently numbered 20 through 24.)

§ 102(e) Rejections – Brown et al. '739

The Examiner has rejected claims 1, 2, 10, 12, 13, 16, 18, 20 and 21 under § 102(e) as being anticipated by Brown et al., US Patent No. 6,201,739 (henceforth, "Brown et al. '739"). The Examiner's rejection is respectfully traversed.

Claim 2 has been canceled, thereby rendering moot the Examiner's rejection of this claim.

Brown et al. '739 teach a flash memory device with a suspend pin for suspending write and erase operations to allow a read operation to take priority over a write or erase operation already in progress. The problem addressed by their device, as described in column 3 lines 36-60, is that of simultaneous management of data storage and direct execution of code on the same flash memory: data storage occasionally needs write and erase operations, which interfere with the immediate access to code that is needed by direct execution. They note the applicability of their device to prior art systems such as the system illustrated in Figure 3, in which code stored in flash memory **104** is first copied to volatile memory **102** and then executed directly by processor **100** from volatile memory **102**. They also note in passing (column 5 line 33) that, although the primary intended application of their invention is to directly executable flash memories, their invention could also be used with flash memories that are not directly executable, for example with NAND flash memoiries.

Strictly speaking, Brown et al. '739 do not literally anticipate the present invention, because their focus is almost entirely on a directly executable replacement of flash memory **104**, and NAND flash memories are mentioned only in passing. Nevertheless, it could be argued that the mention of NAND flash memories in the context of the prior art of Figure 3 is a hint or suggestion that would point one ordinarily skilled in the art in the direction of the present invention.

What renders the present invention patentably distinct from the teachings of Brown et al. '739 is the inclusion of a logic, separate from the processor that executes the code, for copying the code from the flash memory to the volatile memory. Although Brown et al. '739 are silent about how code is copied from flash memory **104** to volatile memory **102**, as best understood, the copying is done by processor **100**. By contrast, in the present invention, as illustrated in Figure **2**, code is copied

from flash memory **14** to S-RAM **40** by logic **42** which is separate from CPU **32** that executes the code. As described on page 10 lines 12-15 of the specification:

> Next, a specific code (set of one or more instructions) is copied automatically from flash memory **14** to S-RAM **40**, without the intervention of CPU **32** (stage 2). For example, such copying of data could optionally be controlled by logic **42**.

There is neither a hint nor a suggestion in Brown et al. '739 of any need for a processor separate from processor **100** for copying code from flash memory **104** to volatile memory **102**.

While continuing to traverse the Examiner's rejections, Applicant has, in order to expedite the prosecution, chosen to amend independent claims 1, 12, 18, 20 and 21 in order to clarify and emphasize the crucial distinctions between the flash memory device of the present invention and the flash memory device of Brown et al. '739. Specifically, claims 1, 12, 18. 20 and 21 have been amended to clarify that the flash memory device of the present invention includes a logic for receiving a command to move at least a portion of the code stored in the flash memory from the flash memory to the volatile memory, and that the logic is separate from the processor that executes the code. In the case of claim 1, this amendment includes incorporating the limitations of claim 2 in claim 1. Consequently, claim 2 has been canceled.

Support for these amendments can be found in the specification. Specifically, support for the inclusion of a logic for receiving a command to move code from the flash memory to the volatile memory is found in claim 2 as filed; and support for this logic being separate from the processor that executes the code is found in Figure 2, which shows logic **42** as separate from CPU **32**.

In addition, for clarity, a comma has been inserted in claim 18(a) to separate the recitation of the flash memory from the recitation of the volatile memory component.

Amended independent claims 1, 12, 18, 20 and 21 now feature language which makes it absolutely clear that the flash memory device of the present invention includes a logic, separate from the processor that executes the code, for copying the code from the flash memory to the volatile memory. Applicant believes that the amendment of the claims completely overcomes the Examiner's rejections on § 102(b) grounds.

With independent claims 1 and 12 allowable in their present form, it follows that claims 10, 13 and 16, that depend therefrom, also are allowable.

§ 103(a) Rejections – Brown et al. '739 and Anderson et al. '577

The Examiner has rejected claims 3-5 under § 103(a) as being unpatentable over Brown et al. '739 and Anderson et al., US Patent No. 6,295,577. The Examiner's rejection is respectfully traversed.

It is demonstrated above that independent claim 1 is allowable in its present form. It follows that claims 3-5, that depend therefrom, also are allowable.

§ 103(a) Rejections – Brown et al. '739 an Mills et al. '688

The Examiner has rejected claims 6 and 7 under § 103(a) as being unpatentable over Brown et al. '739 and Mills et al., US Patent No. 6,385,688. The Examiner's rejection is respectfully traversed.

It is demonstrated above that independent claim 1 is allowable in its present form. It follows that claims 6 and 7, that depend therefrom, also are allowable.

§ 103(a) Rejections – Brown et al. '739 and Nakata '101

The Examiner has rejected claims 8, 9, 14 and 15 under § 103(a) as being unpatentable over Brown et al. '739 and Nakata, US Patent No. 6,523,101. The Examiner's rejection is respectfully traversed.

It is demonstrated above that independent claims 1 and 12 are allowable in their present form. It follows that claims 8, 9, 14 and 15, that depend therefrom, also are allowable.

§ 103(a) Rejections – Brown et al. '739 and Esfahani et al. '695

The Examiner has rejected claims 11, 17, 19, 22 and 23 under § 103(a) as being unpatentable over Brown et al. '739 and Esfahani et al., US Patent No. 6,434,695 (henceforth, "Esfahani et al. '695"). The Examiner's rejection is respectfully traversed.

It is demonstrated above that independent claims 1 and 12 are allowable in their present form. It follows that claims 11 and 17, that depend therefrom, also are allowable.

Esfahani et al. '695 describe what Apple Computer, Inc. did when the "Toolbox ROM" code of its Macintosh computer became too big to fit into the computer's ROM. Only low-level OS code **31** is stored in Boot ROM **11**. The rest of the code that heretofore would have been stored in Boot ROM **11** now is stored elsewhere, in compressed form, and is copied to RAM **12** for execution.

The difference between the present invention and the combined teachings of Brown et al. '739 and Esfahani et al. '695 is captured in the following citation from page 10 lines 15-17 of the specification:

> S-RAM **40** is optionally very small, such that the copied code is preferably only sufficient for permitting the basic initialization of system **30**...

Methodologically, according to the present invention, any of the boot code, including the initially executed boot code for "basic initialization of system **30**", may be stored in the flash memory and copied from the flash memory to the volatile memory for execution. This is in contrast to Esfahani et al. '695, who store only mid-level OS code **32** and high-level OS code **33** outside of Boot ROM **11** and who copy only mid-level OS code **32** and high-level OS code **33** to RAM **12** for execution. There is neither a hint nor a suggestion in Esfahani et al. '695 that low-level OS code **31** could be stored in a mass storage device and copied to a volatile memory such as RAM **12** for execution. Indeed, storing low-level OS code **31** anywhere but Boot ROM **11** would render the computer inoperable because CPU **10** must execute low-level OS code in Boot ROM **11** just to get started.

Structurally, according to the present invention, the volatile memory may be only large enough to contain the bare minimum of boot code needed for basic initialization of get system **30**. By contrast, RAM **12** of Esfahani et al. '695 is a general purpose RAM that must be large enough to store, not only decompressed mid-level OS code **32**, but also high-level OS code **33** and both application code and application data as needed. There is neither a hint nor a suggestion in Esfahani et al. '695 of the feasibility of using a volatile memory only large enough to hold low-level OS code **31** for the in-place execution of low-level OS code **31**.

While continuing to traverse the Examiner's rejections, Applicant has, in order to expedite the prosecution, chosen to amend independent claims 19 and 22 in order to clarify and emphasize the crucial distinctions between the method of the present invention and the combined teachings of Brown et al. '739 and Esfahani et al. '695. Specifically, claims 19 and 22 have been amended to clarify that the code stored in the flash memory is boot code, and that the portion of the boot code that is transferred

to the volatile memory component is for basic initialization of the system. Support for this amendment is found in the specification in the above citation from page 10 lines 15-17.

In addition, because the "first portion of the code" recited in claim 23 is part of the boot code for basic initialization of the system, claim 22 has been amended to include the limitations of claim 23 that refer to the "first portion of the code". Correspondingly, these limitations have been deleted from claim 23. In addition, "copying" in claim 23 has been changed to "transferring", for stylistic consistency.

Amended independent claims 19 and 22 now feature language which makes it absolutely clear that boot code for basic initialization of the system is transferred from the flash memory to the volatile memory. Applicant believes that the amendment of the claims completely overcomes the Examiner's rejections on § 103(a) grounds.

With independent claim 22 allowable in its present form, it follows that claim 23, that depends therefrom, also is allowable.

In addition, new independent claims 24-28 have been added to recite the structural difference between the present invention and the combined teachings of Brown et al. '739 and Esfahani et al. '695. New claims 24-28 are independent claims 1, 12, 18, 20 and 21 as filed, respectively, with the additional limitations that the recited code is boot code, and that the volatile memory component is only large enough to store a portion of the boot code that is sufficient for basic initialization of the system. Support for new claims 24-28 is found in claims 1, 12, 18, 20 and 21 as filed and in the above citation from page 10 lines 15-17 of the specification.

Objections to the Drawings

The Examiner has objected to Figure 1 for lacking a legend indicating that the Figure illustrates prior art. Attached please find a corrected version of Figure 1 including such a legend in red.

In view of the above amendments and remarks it is respectfully submitted that independent claims 1, 12, 18-22 and 24-28, and hence dependent claims 3-11, 13-17 and 23 are in condition for allowance. Prompt notice of allowance is respectfully and earnestly solicited.

Respectfully submitted,

Mark M. Friedman
Attorney for Applicant
Registration No. 33,883

Date: June 30, 2003

VERSION WITH MARKINGS TO SHOW CHANGES MADE

In the claims:

Claims 1, 12 and 18-23 have been amended as follows:

1. (Amended) A flash-based unit for providing code to be executed by an external processor that is in communication with the flash-based unit by a bus, the flash-based unit comprising:

(a) a flash memory for storing the code to be executed, said flash memory being of a type such that the code cannot be executed in place from said flash memory; **[and]**

(b) a volatile memory component for receiving at least a portion of the code to be executed, such that at least said portion of the code is executed by the external processor from said volatile memory component; and

(c) a logic, separate from the external processor, for receiving a command to move said at least portion of the code from said flash memory to said volatile memory component..

12. (Amended) A system for executing code from a restricted non-volatile memory, the restricted non-volatile memory being characterized in that code cannot be directly executed from the restricted non-volatile memory, the system comprising:

(a) a CPU for executing the code;

(b) a volatile memory component in direct communication with the restricted non-volatile memory for holding at least a portion of the

code to be executed, said at least a portion of the code being transferred from the restricted non-volatile memory, such that said CPU executes said at least a portion of the code from said volatile memory component; and

(c) a logic, separate from said CPU, for receiving a command to move said at least portion of the code from the restricted non-volatile memory to said volatile memory component.

18. (Amended) A system for executing code, comprising:

(a) a flash-based unit for storing the code to be executed, said flash-based unit comprising a flash memory of a restricted type, being characterized in that the code cannot be directly executed from said flash memory, and a volatile memory component for receiving a portion of the code to be executed; **[and]**

(b) a processor for executing the code, said processor receiving at least said portion of the code from said volatile memory component; and

(c) a logic, separate from said processor, for receiving a command to move said portion of the code from said flash memory to said volatile memory component;

wherein an additional memory component is not required for executing the code by said processor.

19. (Amended) A method for booting a system, the system featuring a processor for executing boot code, the method comprising:

providing a flash-based unit in the system for storing the boot code to be executed, said flash-based unit comprising a flash memory of a restricted type, being characterized in that code cannot be directly executed from said flash memory, and a volatile memory component for receiving a portion of the boot code to be executed, said portion of the boot code being for basic initialization of the system;

sending a busy signal to said processor;

transferring said portion of the boot code to said volatile memory component;

removing said busy signal; and

executing said portion of the boot code by said processor to boot the system.

20. (Amended) A flash-based unit for providing code to be executed by an external processor, consisting essentially of:

(a) a flash memory for storing the code to be executed, said flash memory being of a type such that the code cannot be executed in place from said flash memory, **[and]**

(b) a volatile memory component for receiving at least a portion of the code to be executed, such that at least said portion of the code is executed by the external processor from said volatile memory component; and

(c) a logic, separate from the external processor, for receiving a command to move said at least portion of the code from said flash memory to said volatile memory component.

21. (Amended) A flash-based unit for providing code to be executed by an external processor, comprising:

(a) a flash memory for storing the code to be executed, said flash memory being of a type such that the external processor cannot read the code to be executed directly from said flash memory; **[and]**

(b) a volatile memory component for receiving at least a portion of the code to be executed, such that at least said portion of the code is executed by the external processor from said volatile memory component; and

(c) a logic, separate from the external processor, for receiving a command to move said at least portion of the code from said flash memory to said volatile memory component.

22. (Amended) A method for booting a system, the system featuring a processor for executing boot code, the method comprising:

providing a flash-based unit in the system for storing the boot code to be executed, said flash-based unit comprising a flash memory of a restricted type, being characterized in that code cannot be directly executed from said flash memory, and a volatile memory component for receiving a portion of the boot code to be executed;

transferring **[said]** a first portion of the boot code to said volatile memory component, said first portion of the boot code being for basic initialization of the system and containing a command for copying a second portion of the code; and

executing said first portion of the boot code by said processor to boot the system.

23. (Amended) The method of claim 22, **[wherein transferring said portion of the code to said volatile memory component further comprises]** further comprising the step of:

[transferring a first portion of the code to said volatile memory component, said first portion of the code containing a command for copying a second portion of the code;

executing said command by said processor; and

copying said] transferring a second portion of the code to said volatile memory component for booting the system.

New claims 24-28 have been added as follows:

24. (New) A flash-based unit for providing boot code to be executed by an external processor, comprising:

(a) a flash memory for storing the boot code to be executed, said flash memory being of a type such that the boot code cannot be executed in place from said flash memory; and

(b) a volatile memory component for receiving at least a portion of the boot code to be executed, such that at least said portion of the boot code is executed by the external processor from said volatile memory component, said at least portion of the boot code being only sufficient for basic initialization of a system that includes the external processor, said volatile memory component being only large enough to store said at least portion of the boot code.

25. (New) A system for executing boot code from a restricted non-volatile memory, the restricted non-volatile memory being characterized in that code cannot be directly executed from the restricted non-volatile memory, the system comprising:

(a) a CPU for executing the boot code; and

(b) a volatile memory component in direct communication with the restricted non-volatile memory for holding at least a portion of the boot code to be executed, said at least portion of the boot code being transferred from the restricted non-volatile memory, such that said CPU executes said at least portion of the boot code from said volatile memory component, said at least portion of the boot code being only sufficient for basic initialization of the system, said volatile memory component being only large enough to store said at least portion of the boot code.

26. (New) A system for executing boot code, comprising:

(a) a flash-based unit for storing the boot code to be executed, said flash-based unit comprising a flash memory of a restricted type, being characterized in that the boot code cannot be directly executed from said flash memory, and a volatile memory component for receiving a portion of the boot code to be executed, said portion of the boot code being only sufficient for basic initialization of the system, said volatile memory component being only large enough to store said at least portion of the boot code; and

(b) a processor for executing the boot code, said processor receiving at least said portion of the boot code from said volatile memory component;

wherein an additional memory component is not required for executing the boot code by said processor.

27. (New) A flash-based unit for providing boot code to be executed by an external processor, consisting essentially of:

(a) a flash memory for storing the boot code to be executed, said flash memory being of a type such that the boot code cannot be executed in place from said flash memory, and

(b) a volatile memory component for receiving at least a portion of the boot code to be executed, such that at least said portion of the boot code is executed by the external processor from said volatile memory component, said at least portion of the boot code being only sufficient for basic initialization of a system that includes the external processor, said volatile memory component being only large enough to store said at least portion of the boot code.

28. (New) A flash-based unit for providing boot code to be executed by an external processor, comprising:

(a) a flash memory for storing the boot code to be executed, said flash memory being of a type such that the external processor cannot read the boot code to be executed directly from said flash memory; and

(b) a volatile memory component for receiving at least a portion of the boot code to be executed, such that at least said portion of the boot code is executed by the external processor from said volatile memory component, said at least portion of the boot code being only sufficient for basic initialization of a system that includes the external processor, said volatile memory component being only large enough to store said at least portion of the boot code.